Entwurfsdokumentation

VOTEsso

Bürgerbeteiligung Kiel

-

Softwareprojekt Sommer 2022

Gruppe LMS8 EG 016

VOTEsso

Bürgerbeteiligung

Heiko Bielfeldt
Markus Glaubitz
Niclas Nebelung
Sören Petersen
Arne Seufert
Jonas Struve
Jon Stührwoldt
Arne Wiese

4. September 2022

Inhaltsverzeichnis

Kapitel 1

Einleitung

Nachdem die Anforderungen an die Software, das Anwendungsgebiet und die Zielgruppe bestimmt wurden, wird nun ein abstrakteres (und deutlich technischeres) Modell dokumentiert. Die nach dem Pflichtenheft zu erstellende Entwurfsdokumententation legt technische Spezifikationen des zukünfigten Systems der Software fest.

In dieser Phase der Dokumentation sollen erste Nachweise geschaffen werden, die die technische Umsetzbarkeit des Systems und die richtige Funktionsweise zeigen. Wichtig ist hierbei zu beachten, dass es sich um einen ersten Entwurf der Software nach jetzigem Verständnis der verwendeten Technologien und Systemumgebung handelt. Während der Softwareentwicklung werden sich Teile des Systementwurfs nach Notwendigkeiten ändern. Dies wird seperat dokumentiert.

Hier wird versucht die realen Anforderungen in technischer Hinsicht abzubilden. Im Vordergrund steht die Transparenz der Systemarchitektur durch die Abbildung wichtiger Prozesse und Interaktionen zwischen Systemteilen. Um eine möglichst genauen und verständlichen Überblick zu schaffen, werden hier einige verschiedene Diagrammarten, welche sich an der UML orientieren, verwendet:
Der Einsatz von Hardware und die dort eingesetzten Softwareumgebungen zur Ausführung der Systemteile sind in Verteilungsdiagrammen zu erkennen.
Eine kompaktere und weniger detailtierte Ansicht des Systems sind in Komponentendiagrammen zu finden, in denen auch externe Anbindungen realisiert sind.
Sinn der Klassendiagramme ist unter anderem die Zusammenhänge von Klassen und Komponenten darzustellen und die Komplexität dieser gezielt zum Beispiel durch Modularisierung zu reduzieren.
Um die Abläufe und Kommunikation zwischen Komponenten und Klassen zur Laufzeit darzustellen, werden Sequenzdiagramme verwendet.

> Die Diagramme sind nochmal separat in einem Unterordner (als Vektorgrafik) zu finden, da sie zum Teil komplex und recht groß ausfallen und möglicherweise in diesem Dokument schwer zu lesen sind.

1.1 Entwicklungsumgebung

Software	Version	URL
Java Development Kit	17.0.1	`http://www.oracle.com/technetwork/java/javase/downloads/index.html`
Android Studio Chipmunk	2021.2.1 (Patch 2)	`https://developer.android.com/studio`
Android	11	`https://developer.android.com/about/versions/11/`
Spring Boot	2.7.0	`https://spring.io/`
Thymeleaf	von Spring eingebunden	`https://www.thymeleaf.org/`
Bootstrap	von Spring eingebunden	`https://getbootstrap.com/`
Lombok	6.4.3	`https://projectlombok.org/`
Gradle	7.5.1	`https://gradle.org/`
Docker	20.10.17	`https://www.docker.com/`
Git	2.37.3	`https://git-scm.com/`
Retrofit	2.9.0	`https://square.github.io/retrofit/`
PostgreSQL	9.6.21	`https://www.postgresql.org/`

Tabelle 1.1: Enwicklungsumgebung

Kapitel 2

Team-Aufteilung

Name	Zuständigkeit
Markus Glaubitz	App
Niclas Nebelung	App
Sören Petersen	App
Arne Seufert	App
Heiko Bielfeldt	Frontend Web, Backend
Jonas Struve	Backend
Jon Stührwoldt	Backend
Arne Wiese	Backend

Aufgrund des derzeitigen Projektfortschrittes ist bis jetzt noch keine genauere Aufgabenteilung erfolgt. Diese wird in der Entwicklungszeit dynamisch stattfinden.

Kapitel 3

Komponentendiagramme

3.1 Komponentendiagramm Web

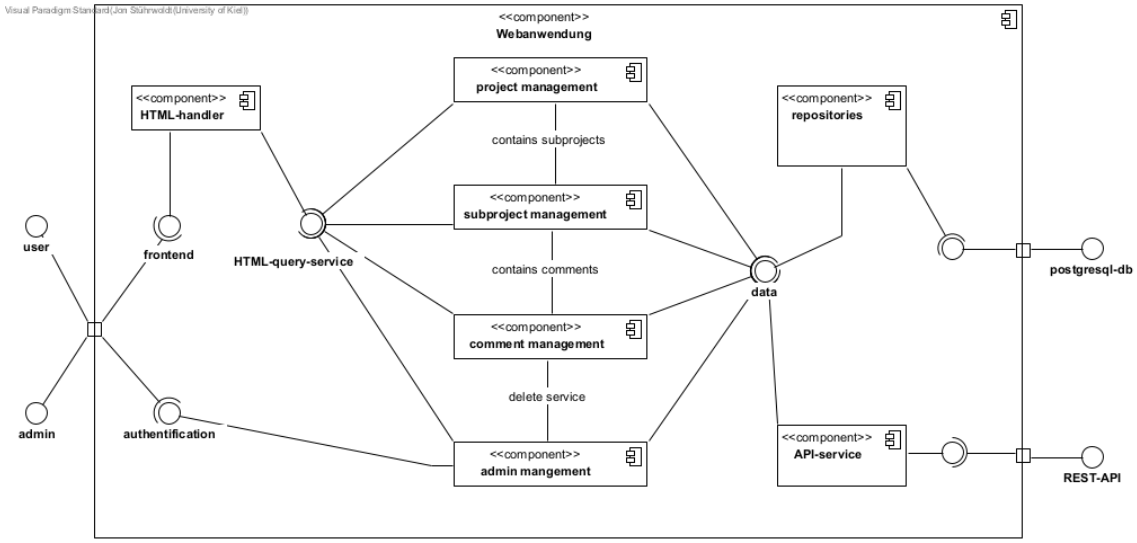

Abbildung 3.1: Komponentendiagramm - Web

Subkomponente	Aufgabe
HTML-handler	Verwaltet und erstellt HTML-Dokumente und deren Inhalt.
project management	Ruft Daten für Projekte auf und stellt sie dem jeweiligen HTML-Dokument bereit.
subproject management	Ruft Daten für Teilprojekte auf und stellt sie dem jeweiligen HTML-Dokument bereit.
comment management	Ruft Daten für Kommentare auf und stellt sie dem jeweiligen HTML-Dokument (Detailübersicht eines Teilprojekts) bereit. Verwaltet des Weiteren den Löschvorgang von Kommentaren für Administratoren.
admin management	Ruft Daten für die Anmeldung als Administrator auf und stellt sie dem jeweiligen HTML-Dokument bereit. Verwaltet außerdem die Autorisierung zum Löschen von Kommentaren.
repositories	Stellt den angefragten Datenbestand der PostgreSQL-Datenbank bereit und ist zentral für die nachträgliche Aktualisierung der Daten im Falle einer Änderung (z.B. Kommentar erstellen oder löschen).
API-services	Verwaltet den Datenfluss zwischen Backend und App.

Tabelle 3.1: Komponentenbeschreibung - Web

3.2 Komponentendiagramm App

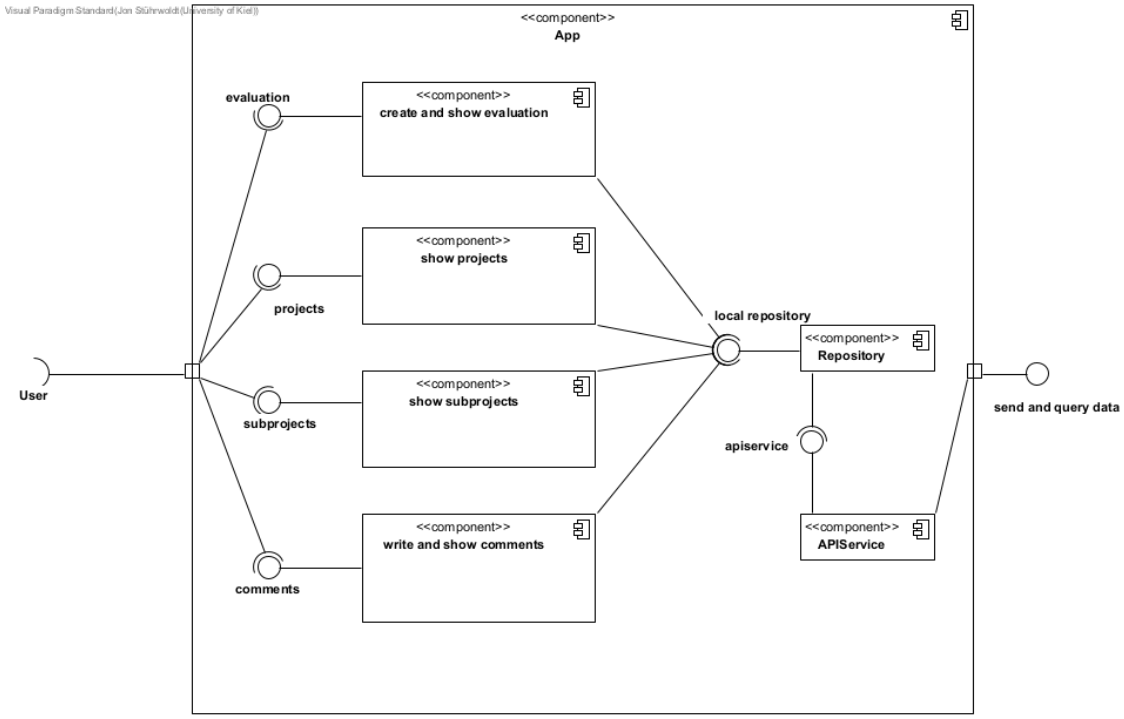

Abbildung 3.2: Komponentendiagramm - App

Subkomponente	Aufgabe
write and show comments	Diese Komponente ermöglicht es dem User zu vorher ausgewählten Teilprojekten einen für alle anderen User sichtbaren Kommentar abzugeben oder Kommentare von anderen Usern anzuschauen.
create and show evaluation	Diese Komponente bietet dem User die Möglichkeit eine Bewertung in Form von Daumen hoch oder Daumen runter abzugeben und die Bewertung anderer User anzuschauen.
Repository	Diese Komponente speichert die benötigten Daten zur Laufzeit und stellt sie den anderen Komponenten zur Verfügung. Außerdem kommuniziert sie mit dem APIService um Daten zu senden und zu empfangen.
show projects	In dieser Komponente werden dem User alle verfügbaren Hauptprojekte angezeigt. Außerdem ist sie dafür verantwortlich nach Auswahl eines Hauptprojektes die entsprechende Detailseite anzuzeigen.
show subprojects	In dieser Komponente werden dem User alle zu einem Hauptprojekt zugehörigen Teilprojekte, nach Entfernung sortiert, angezeigt. Zudem sorgt diese Komponente auch dafür, die entsprechende Teilprojekt Detailseite anzuzeigen.
APIService	Diese Komponente übernimmt die Kommunikation mit dem Backend. Sie ist sowohl dafür verantwortlich beim Start der App die Daten zu holen, als auch neue Daten zu senden.
API-services	Verwaltet den Datenfluss zwischen Backend und App.

Tabelle 3.2: Komponentenbeschreibung - App

Kapitel 4

Verteilungsdiagramm

Da das zukünftige Deployment des Systems noch durchaus ungewiss ist, ist hier wie abgesprochen das Deployment in der Entwicklungsphase an der CAU dokumentiert.

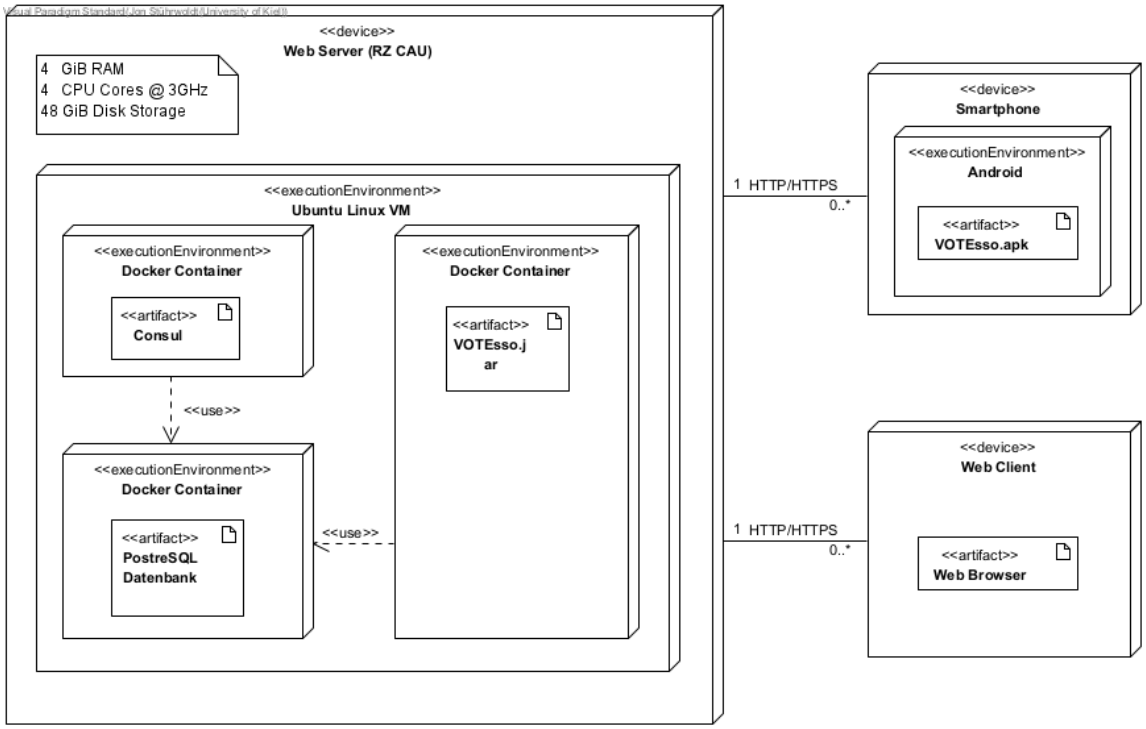

Abbildung 4.1: Verteilungsdiagramm

Kapitel 5

Klassendiagramme

5.1 Klassendiagramme Web

5.1.1 VOTEsso und Configurations

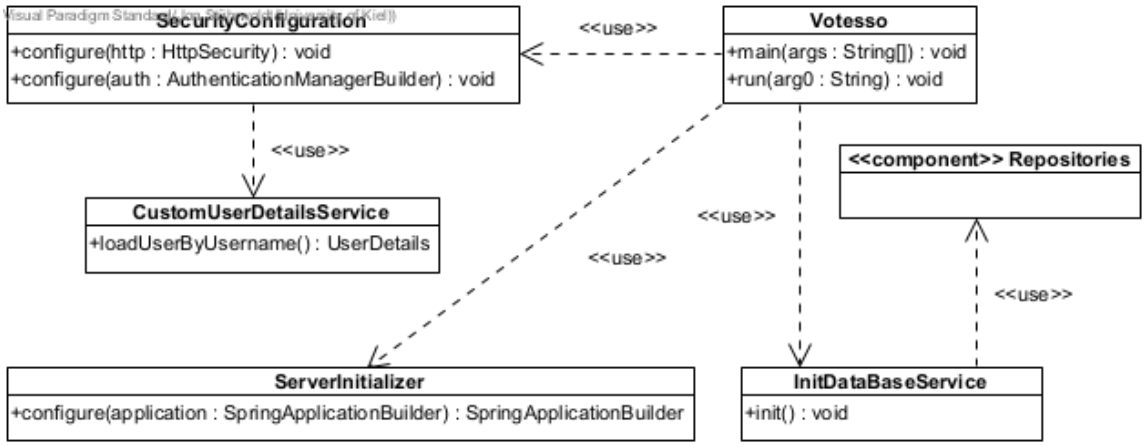

Abbildung 5.1: Klassendiagramm - Configurations

Klassenname	Aufgabe
Votesso	Main-Klasse, über die das Programm gestartet werden kann.
SecurityConfiguration	Verwaltet den Zugriff auf URLs.
CustomUserDetailsService	Kümmert sich um die Autorisierung, beim Login von Administratoren.
ServerInitializer	Konfiguriert und baut die Anwendung.
InitDataBaseService	Initialisiert alle Repositries mit dem jeweiligen Datenbankinhalt.

Tabelle 5.1: Klassenbeschreibung - Web, Configurations

5.1.2 Model

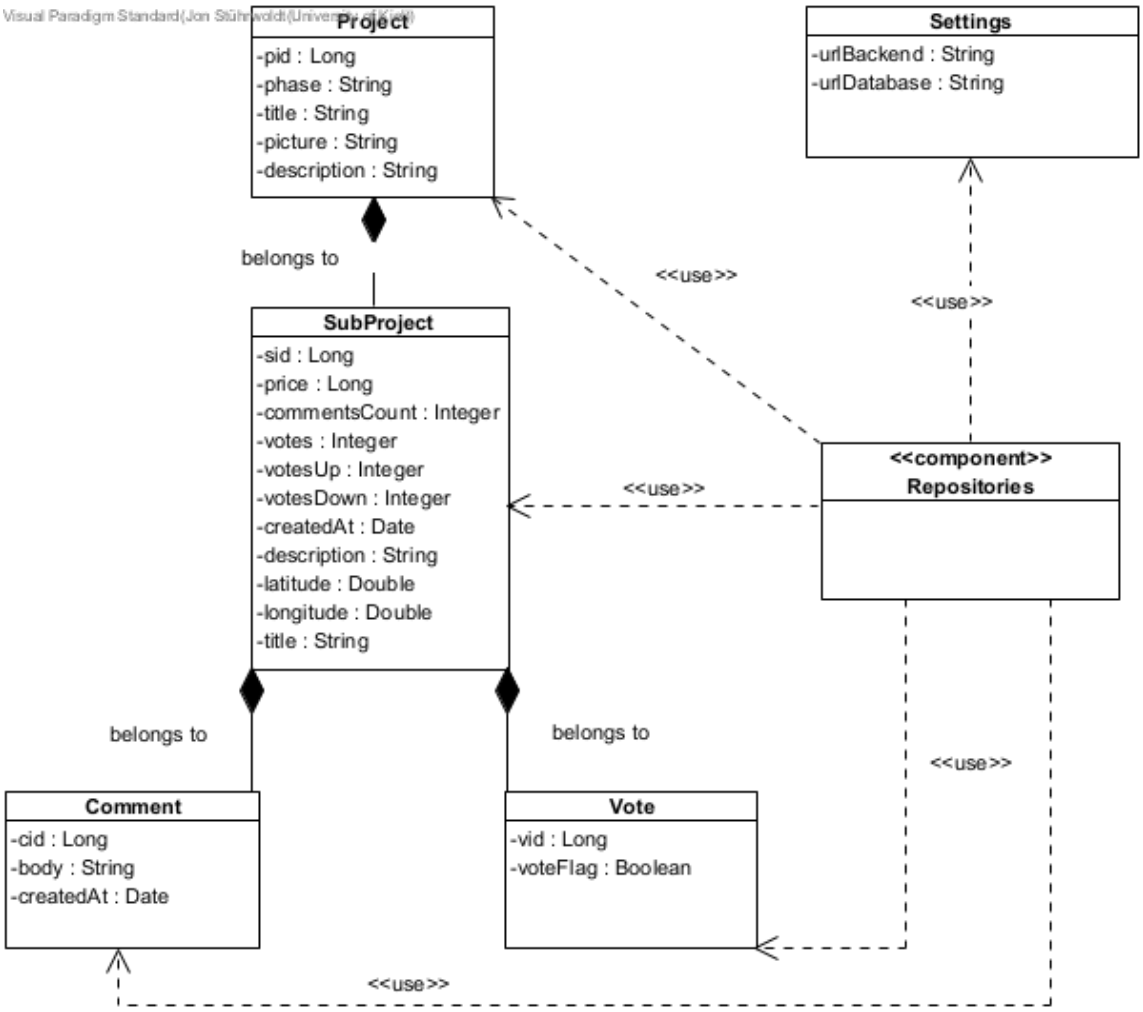

Abbildung 5.2: Klassendiagramm - Model

Klassenname	Aufgabe
Project	Klasse, die den Datentypen Project repräsentiert und alle nötigen Informationen eines (Haupt-)Projekts enthält. Der Aufbau der Klasse dient als Vorlage für die Objekte, die im zugehörigen Repository abgespeichert werden.
SubProject	Klasse, die den Datentypen SubProject repräsentiert und alle nötigen Informationen eines TeilProjekts enthält. Der Aufbau der Klasse dient als Vorlage für die Objekte, die im zugehörigen Repository abgespeichert werden.
Comment	Klasse, die den Datentypen Comment repräsentiert und alle nötigen Informationen eines Kommentars enthält. Der Aufbau der Klasse dient als Vorlage für die Objekte, die im zugehörigen Repository abgespeichert werden.
Vote	Klasse, die den Datentypen Vote repräsentiert und alle nötigen Informationen einer Abstimmung/Bewertung enthält. Der Aufbau der Klasse dient als Vorlage für die Objekte, die im zugehörigen Repository abgespeichert werden.
Settings	Klasse, die die URLs enthält, welche durch den Administrator konfigurierbar sind. Von dieser Klasse soll später lediglich eine Instanz existieren. Der Aufbau der Klasse dient als Vorlage für das eine Objekt, welches im zugehörigen Repository abgespeichert wird.

Tabelle 5.2: Klassenbeschreibung - Web, Model

5.1.3 Repositories

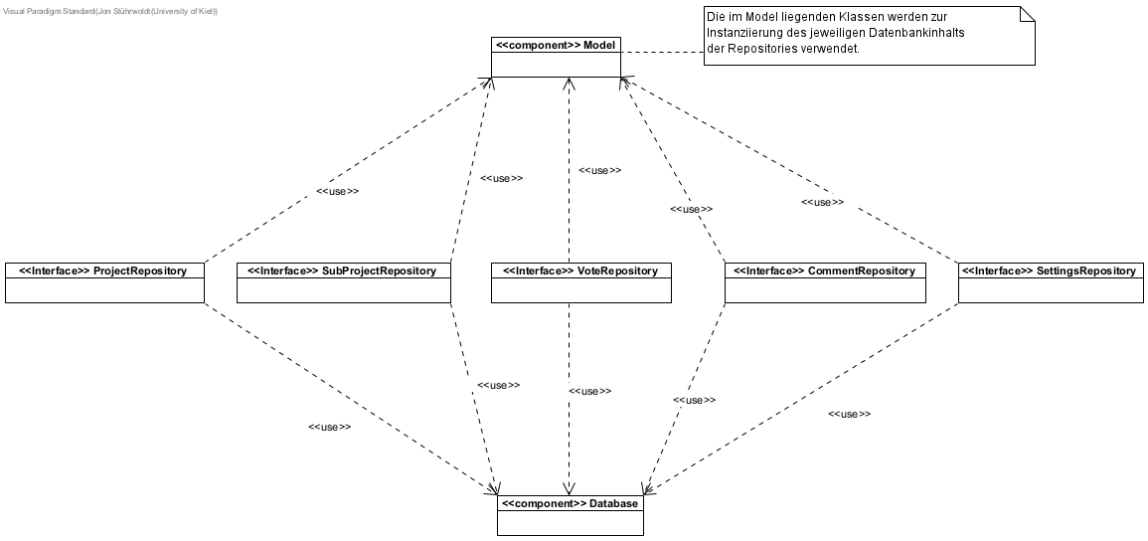

Abbildung 5.3: Klassendiagramm - Repositories

Klassenname	Aufgabe
ProjectRepository	Interface, welches die Verwaltung von Objekten des Datentyps Project übernimmt.
SubProjectRepository	Interface, welches die Verwaltung von Objekten des Datentyps SubProject übernimmt.
VoteRepository	Interface, welches die Verwaltung von Objekten des Datentyps Vote übernimmt.
CommentRepository	Interface, welches die Verwaltung von Objekten des Datentyps Comment übernimmt.
SettingsRepository	Interface, welches die Verwaltung der einen Instanz der Klasse Settings übernimmt.

Tabelle 5.3: Klassenbeschreibung - Web, Repositories

Die Interfaces werden durch das Spring Boot Framework realisiert.

5.1.4 Controller

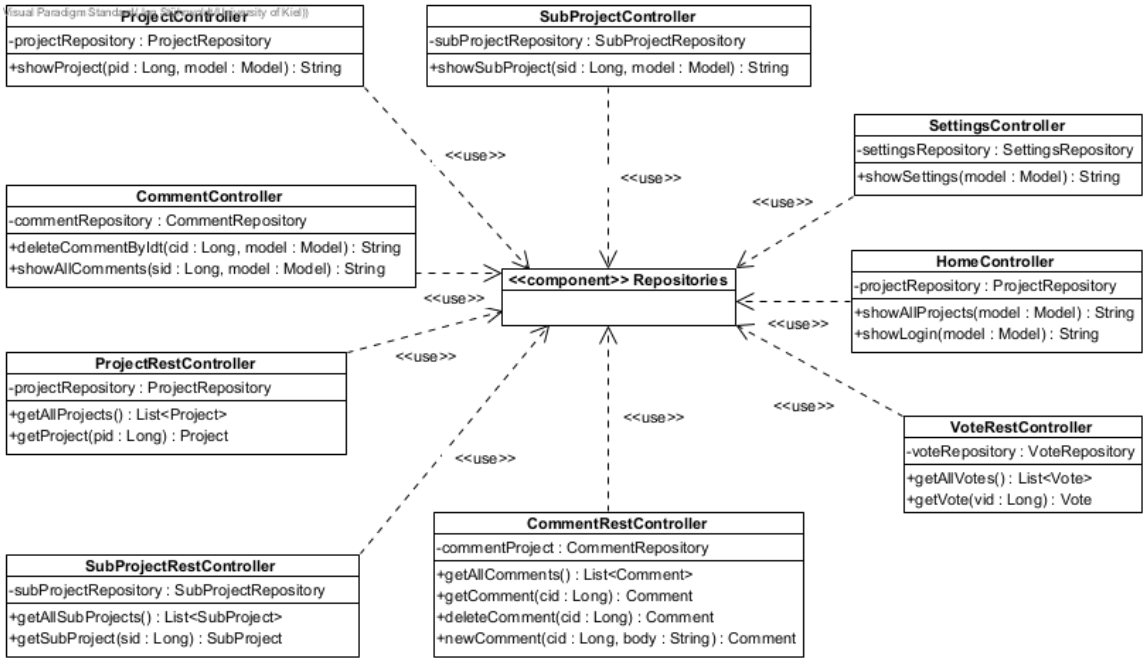

Abbildung 5.4: Klassendiagramm - Controller

Klassenname	Aufgabe
HomeController	Klasse, die eine HTML-Seite zurückgibt, die eine Liste aller (Haupt-)Projekte enthält und dem Administrator die Möglichkeit bietet, über einen Button auf die Login-Seite weitergeleitet zu werden.
ProjectController	Klasse, die eine HTML-Seite zurückgibt, welche alle Informationen eines Projektes enthält. Zu diesen Informationen zählt unter anderem eine Liste der zugehörigen Teilprojekte, in der die jeweiligen Seiten der Teilprojekte verinkt sind.
SubProjectController	Klasse, die eine HTML-Seite zurückgibt, die alle Informationen eines Teilprojektes und einen Link zu den Kommentaren dieses Teilprojektes enthält.
CommentController	Klasse, die eine HTML-Seite zurückgibt, die alle Kommentare eines gegebenen Teilprojektes enthält. Administratoren haben auf dieser Seite außerdem die Möglichkeit Kommentare zu löschen.
SettingsController	Klasse, die eine HTML-Seite zurückgibt, auf welcher Administratoren die System-URLs konfigurieren können.
ProjectRestController	Klasse, die Methoden für den Austausch von Projekten mit der App über die Rest-API bereitstellt.
SubProjectRestController	Klasse, die Methoden für den Austausch von Teilprojekten mit der App über die Rest-API bereitstellt.
CommentRestController	Klasse, die Methoden für den Austausch von Kommentaren mit der App über die Rest-API bereitstellt.
VotesRestController	Klasse, die Methoden für den Austausch von Bewertungen/Abstimmungen mit der App über die Rest-API bereitstellt.

Tabelle 5.4: Klassenbeschreibung - Web, Controller

5.2 Klassendiagramm App

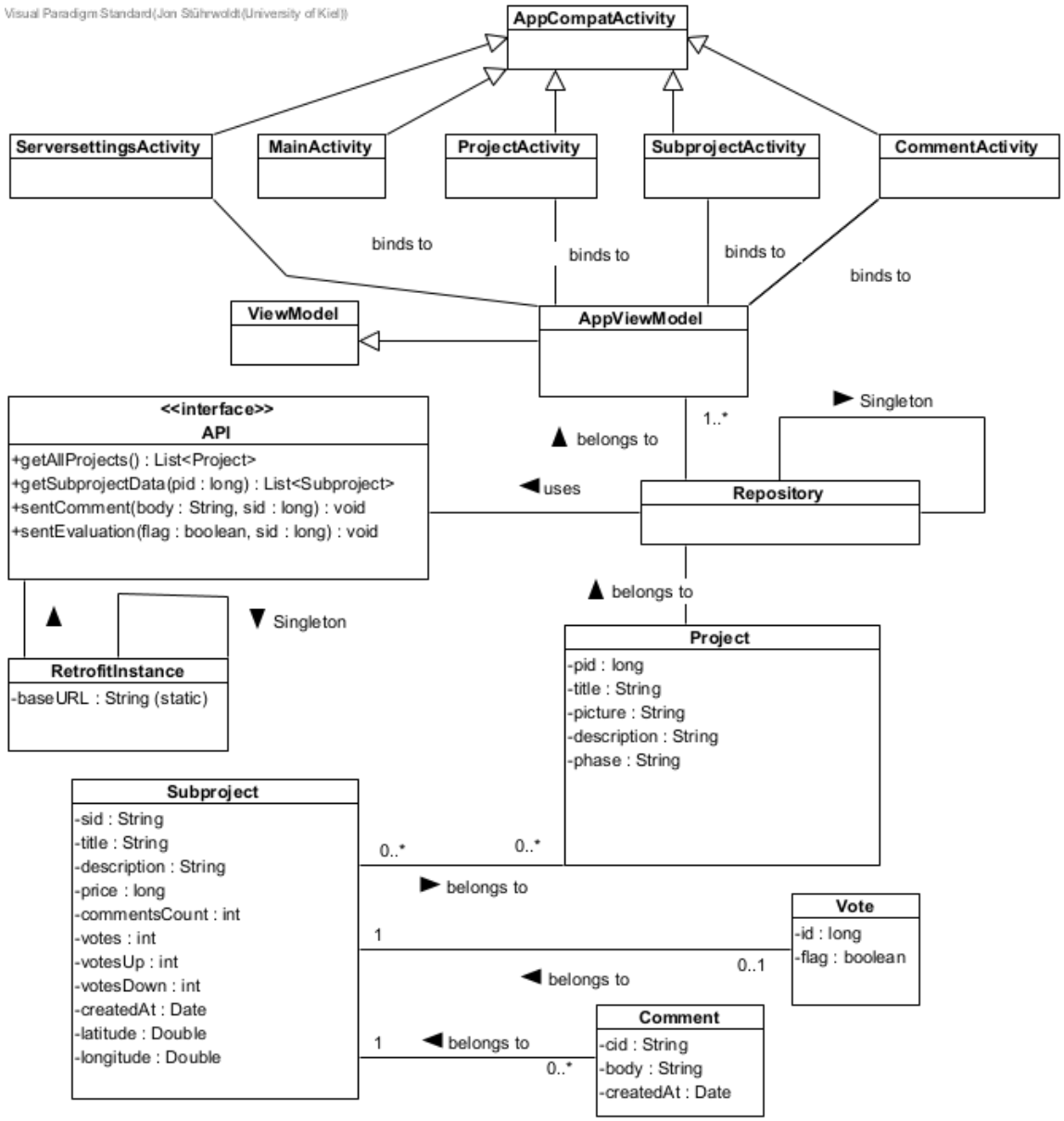

Abbildung 5.5: Klassendiagramm - App

Klassenname	Aufgabe
AppCompatActivity	Von Android bereitgestellte Basisklasse für Activities.
MainActivity	Realisiert die Projektübersichtsseite, Startpunkt unserer App.
Serversettingsactiivity	Realisiert die Server-Einstellungsseite. Hier kann die URL des Backend angepasst werden.
ProjectActivity	Realisiert die Projektdetailseite und die Auflistung der Teilprojekte.
SubprojectActivity	Realisiert die Teilprojektseite mit einem Kommentareingabebereich.
CommentActivity	Realisiert die Auflistung aller zu dem Teilprojekt geschriebenen Kommentare.
AppViewModel	Stellt das Bindeglied zwischen View und Model dar. Verarbeitet die vom Repository erhaltenen Daten und stellt es der View bereit.
Viewmodel	Von Android bereitgestellte Basisklasse für unser AppViewModel.
Repository	Ist für den Datenaustausch zwischen App und Backend zuständig. Die Repository-Klasse tätigt sämtliche API-Calls.
RetrofitInstance	Klasse, um API-Calls mittels http zu realisieren.
APIService	Benötigtes Interface für Aufrufe an das Backend (nötig für Retrofit).
Project	Klasse für das Projekt.
Subproject	Klasse für das Teilprojekt.
Comment	Klasse für den Kommentar.
Vote	Klasse, die die Evaluierung der Teilprojekte erleichtert.

Tabelle 5.5: Klassenbeschreibung - App

Kapitel 6

Sequenzdiagramme

6.1 Sequenzdiagramme Web

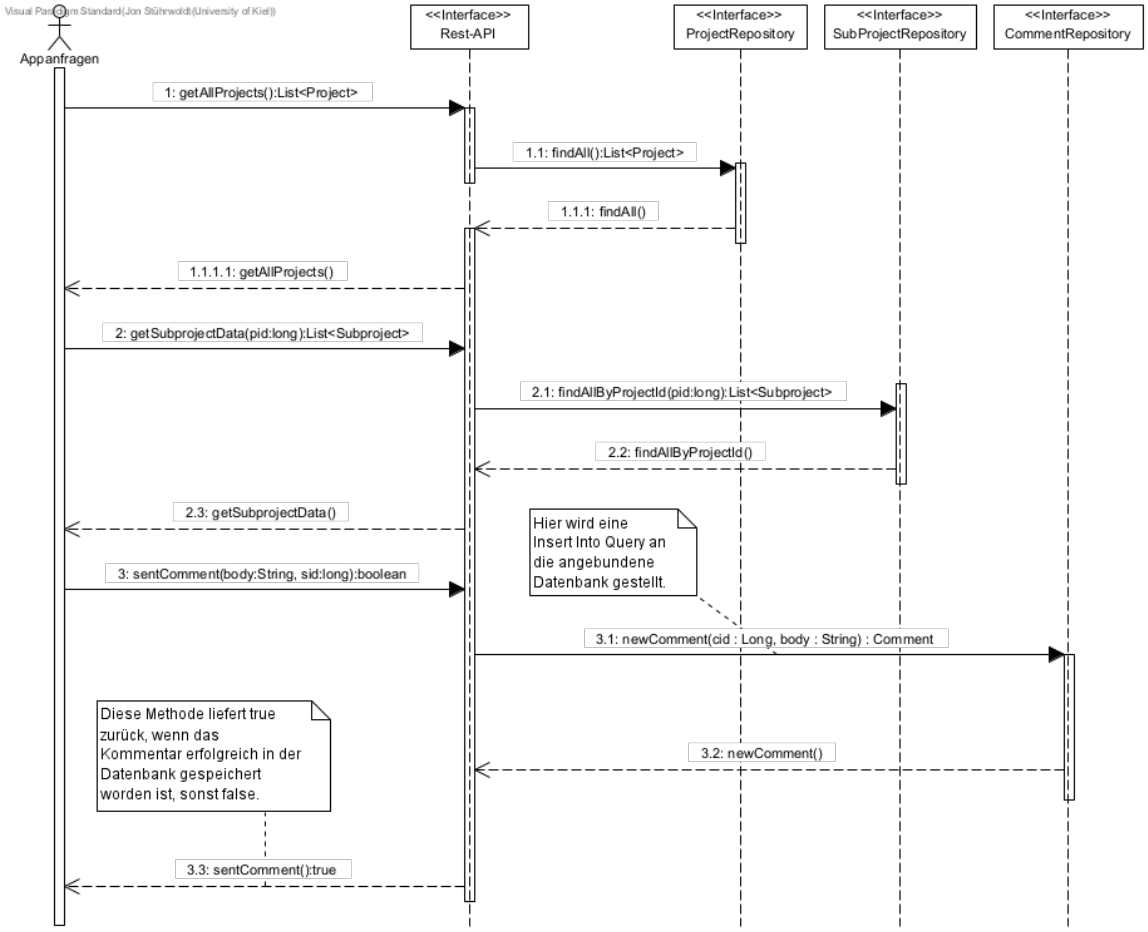

Abbildung 6.1: Sequenzdiagramm - Kommentar schreiben

Dieses Sequenzdiagramm zeigt den zeitlichen Ablauf des Anwendungsfalles "Kommentar bei Teilprojekt abgeben". Die App fragt hierbei beim Backend zuerst die Projektdaten, dann die Teilprojektdaten und schließlich die jeweiligen Kommentarlisten der Teilprojekte an. Die einzelnen Anfragen werden dabei über eine Rest-API geregelt, welches dann im Backend die zur Anfrage gehörigen Daten aus den Repositories abfragt und anschließend wieder an die App zurücksendet.

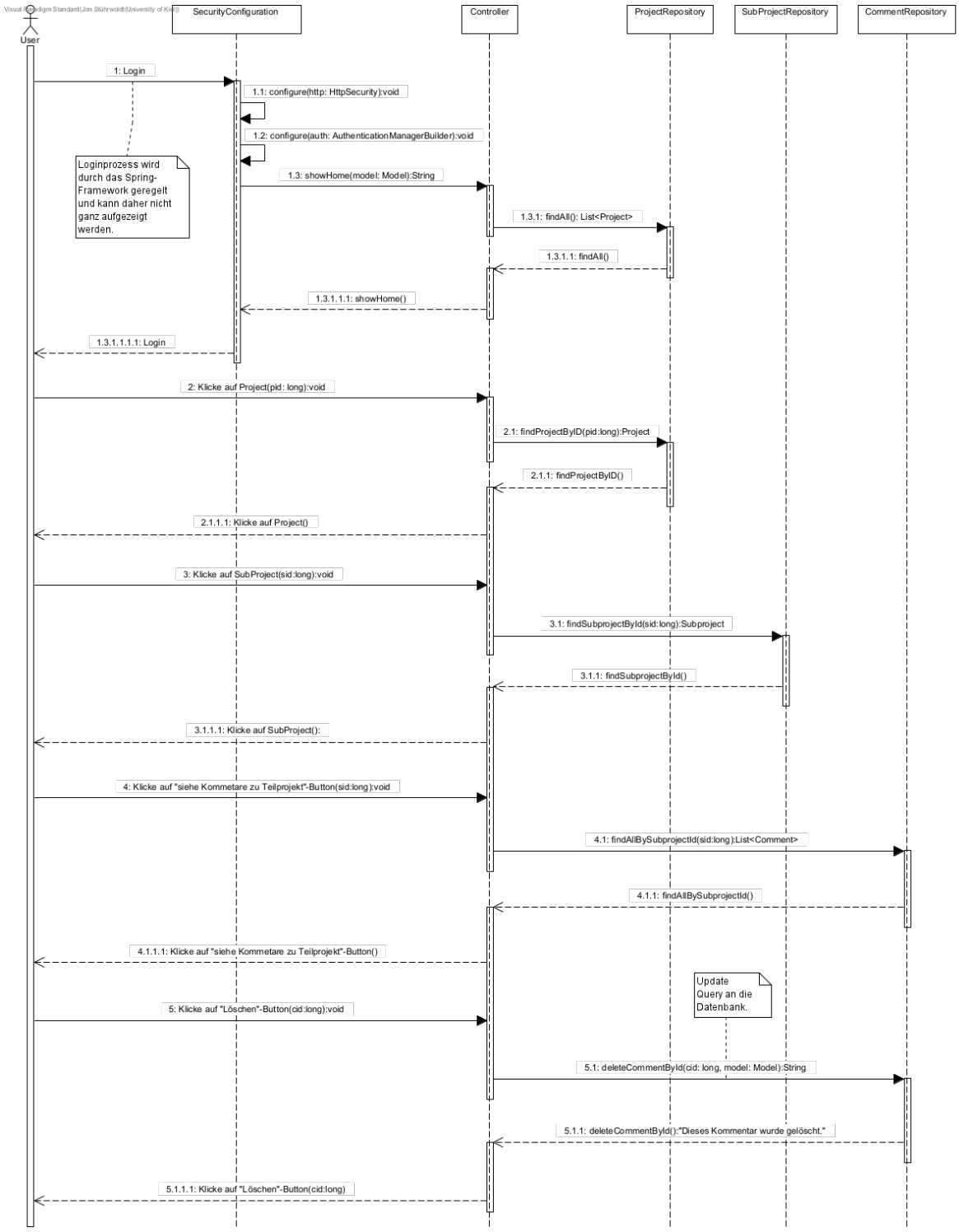

Abbildung 6.2: Sequenzdiagramm - Kommentar löschen

Dieses Sequenzdiagramm zeigt den zeitlichen Ablauf des Anwendungsfalles "Kommentar löschen". Um die Berechtigung zu haben, Kommentare löschen zu können, muss sich der User als Admin einloggen. Der Login-Prozess wird vom Spring Boot Framework übernommen. Nach einem erfolgreichen Login wird man dann zur "Home"-Seite weitergeleitet. Dabei wird die Methode showHome() vom Controller aufgerufen, der sich erst die nötigen Daten aus dem jeweiligen Repository zieht und anschließend eine auf den Daten basierende .html Datei zurückgibt. Der Ablauf vom Controller bis zur .html Datei geschieht analog mit den Daten vom Projekt, Teilprojekt und Kommentar.

6.2 Sequenzdiagramme App

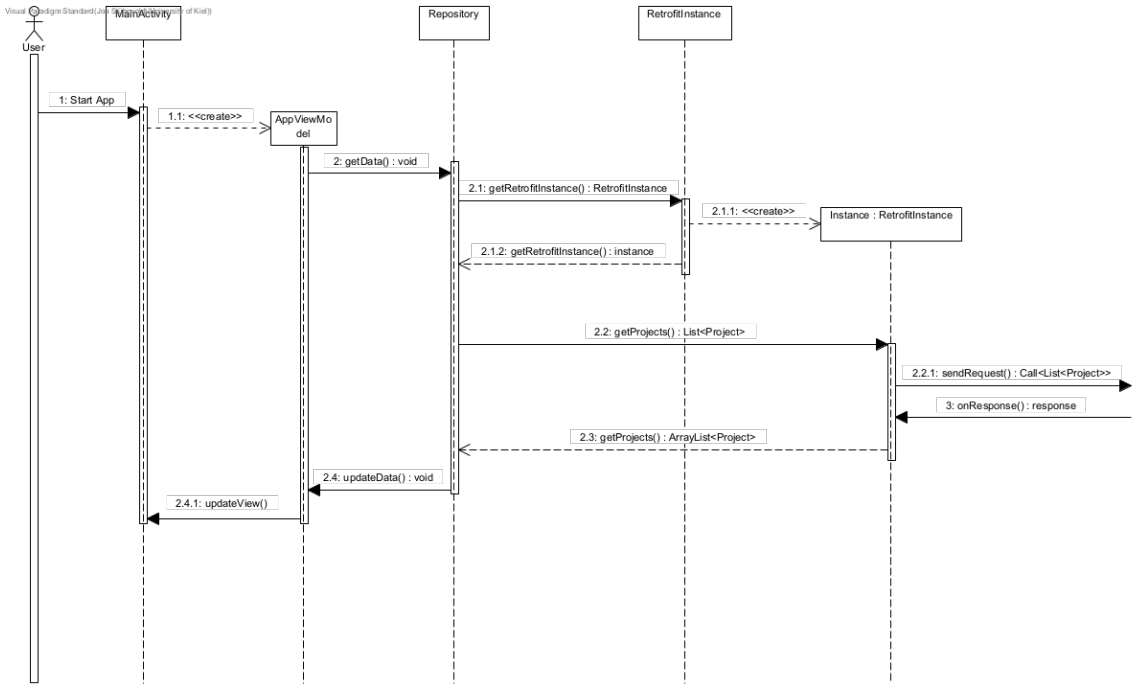

Abbildung 6.3: Sequenzdiagramm - App starten

Dieses Sequenz-Diagramm beschreibt den App-Aufruf und das Afragen der Projektdaten über das Backend. Nach dem Start der App initialisiert die MainActivity das AppViewModel. Das AppViewModel stellt eine Anfrage an das Repository. Dort wird eine RetrofitInstance erstellt, über welche die Projektdaten vom Backend abgefragt werden. Die Daten werden zum AppViewModel zurückgegeben und auf der MainActivity dargestellt.

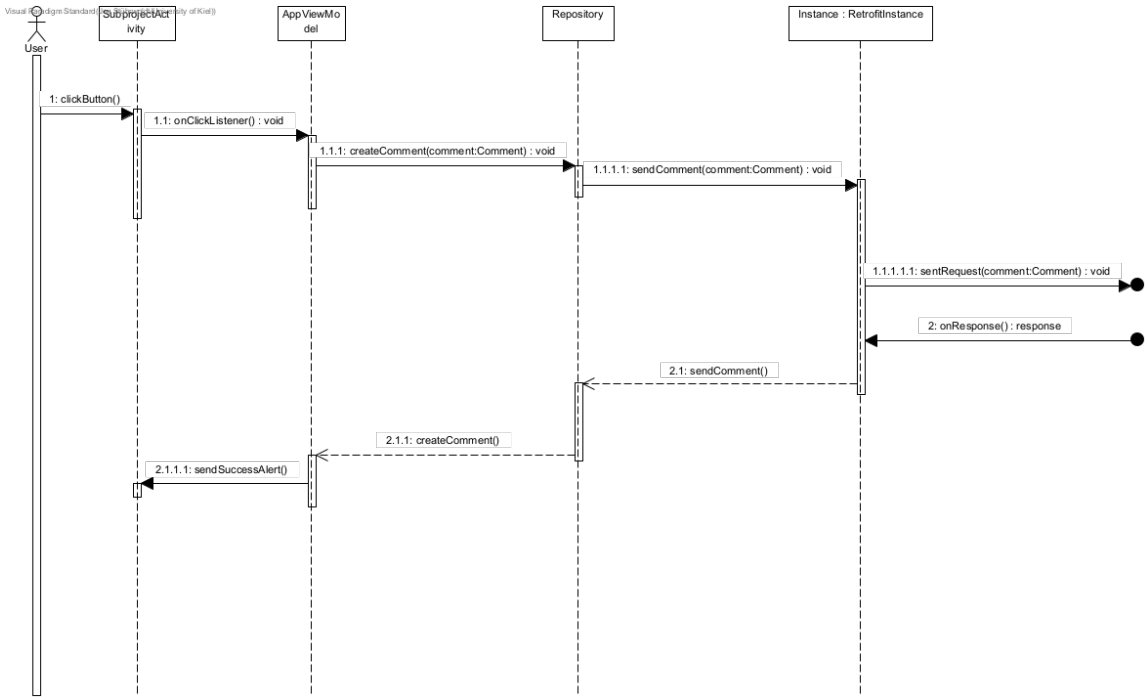

Abbildung 6.4: Sequenzdiagramm - Kommentar schreiben

Die Abbildung zeigt das Sequenz-Diagramm zum Hinzufügen eines neuen Kommentars zu einem Teilprojekt. Nach dem Eingeben des Kommentars kann er über einen Button die Eingabe bestätigen. Dabei ruft das AppViewModel eine Methode zum Erstellen eines neuen Kommentars im Repository auf. Das Repository leitet den Kommentar weiter an das Backend. Im Anschluss wird eine Success-Message an den Benutzer zurückgegeben.